ARMY RESEARCH LABORATORY

Diskless Linux Cluster How-To

by Justin L. Shumaker

ARL-TR-3607 **September 2005**

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

Army Research Laboratory
Aberdeen Proving Ground, MD 21005-5068

ARL-TR-3607 **September 2005**

Diskless Linux Cluster How-To

Justin L. Shumaker
Survivability/Lethality Analysis Directorate, ARL

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
September 2005	Final	9 October 2004–22 February 2005

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Diskless Linux Cluster How-To	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)	5d. PROJECT NUMBER
Justin L. Shumaker	1L162618AH80
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
U.S. Army Research Laboratory ATTN: AMSRD-ARL-SL-BE Aberdeen Proving Ground, MD 21005-5068	ARL-TR-3607

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Diskless linux clustering is not yet a turn-key solution. The process of configuring a cluster of diskless linux machines requires many modifications to the stock linux operating system before they can boot cleanly. This guide will help the experienced linux user to take a set of Pre-eXecution Environment capable machines and configure them appropriately.

15. SUBJECT TERMS

diskless, linux cluster, PXE

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Justin L. Shumaker
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL	28	19b. TELEPHONE NUMBER (*Include area code*) 410-278-2834

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

INTENTIONALLY LEFT BLANK.

1. Introduction

The following is a guide designed for instructing a skilled administrator on the steps necessary for configuring a diskless linux cluster. It is assumed that the reader is familiar with TFTP, DHCP, Pre-eXecution boot Environment (PXE), configuring and compiling kernels, and has general unix knowledge. Be prepared for some trial and error. During the writing of this report, I rebooted a diskless node well over 100 times in order to perfect the configuration. With the aid of this report one should only have to reboot their cluster a few times until the configuration is in good working order. As an overview, one will begin by configuring a “master” node from where the “slave” nodes will retrieve their data. First, the process will begin with the configuration of TFTP and DHCP, which will allow the slaves to get an IP, hostname, kernel, and ramdisk. Next, a file from the SysLinux UNIX tape archive is used to permit PXE booting. Later, a custom kernel and ramdisk are created exclusively for the slave nodes for their boot process. Finally, the slaves are configured to operate in a diskless environment. The Post-Install section provides insight into some additional useful configurations as well as how a few of the problems that were encountered along the way were resolved.

There are a number of websites devoted to various methods of configuring a cluster to boot linux in a diskless fashion. The following is a list of websites found to be useful. They may serve as a reference to the information provided in this document:

1. http://frank.harvard.edu/~coldwell/diskless/
2. http://www.linuxforum.com/linux-network-boot.php
3. http://www.linuxnetmag.com/en/issue5/m5diskless1.html

2. Configuration

2.1 Step 1: TFTP

Most linux systems come with a /tftpboot/ directory; if there is not one, create one. Make sure that the service daemon such as xinetd, which is being used to handle services, has the TFTP service enabled. Redhat 9 has xinetd configuration data under "/etc/xinetd.d/". Edit the "tftp" file in this directory and change "disable = yes" to "disable = no". At this point the machine should be capable of handling TFTP connections. If TFTP is not working, consult your TFTP manual and "tail" your "/var/log/messages" to see if there are any obvious errors.

2.2 Step 2: DHCP

When the diskless nodes boot they will try to connect to the master using their PXE to obtain an IP and fetch some data. Configuring DHCP will point them to the necessary data, which will be transferred via TFTP. Make sure that "dhcpd" is installed. DHCP-3 (the default with redhat 9) was used for this configuration. Edit the file "/etc/dhcpd.conf" and use the following as a configuration template:

```
---------------------------------------------------------------
ddns-update-style ad-hoc;
subnet 192.168.1.0 netmask 255.255.255.0 {
group {
  use-host-decl-names on;
  filename "pxelinux.0";
  host n1 {
    hardware ethernet 00:02:B3:52:3B:28;
    fixed-address n1.myhost.com;
  }
  host n2 {
    hardware ethernet 00:02:B3:52:3B:1F;
    fixed-address n2.myhost.com;
  }
}
default-lease-time 86400;
max-lease-time 86400;
option routers 192.168.1.254;
option ip-forwarding off;
---------------------------------------------------------------
```

If you have worked with DHCP before, then the previous configuration syntax should be familiar. If you are unfamiliar with the syntax, then consult the DHCP manual. Adjust the configuration for your custom network configuration (i.e., subnet, number of nodes, etc). After editing is complete, be sure to restart the DHCP daemon "dhcpd", i.e., “ps ax | grep dhcp”. If “dhcpd” is not running, start it and make sure it's using this configuration file (it should be by default). Once again, check "/var/log/messages" to resolve any problems.

2.3 Step 3: SysLinux

Now that TFTP and DHCP are configured and working properly, the diskless nodes can get an IP when booted. The next step they will try to perform is obtaining a file called pxelinux.0 from the DHCP daemon (as indicated by the DHCP configuration file). This file will allow the nodes to get a kernel and ramdisk image, which will be discussed later. Obtain the latest version of syslinux from kernel.org at http://www.kernel.org/pub/linux/utils/boot/syslinux/.

The file you want from the tarball is called "pxelinux.0". Create a directory under your "/tftpboot" directory called "pxelinux.cfg" so you have "/tftpboot/pxelinux.cfg/". Drop the "pxelinux.0" file from the syslinux package into "/tftpboot/". Edit a new file called "default" under "/tftpboot/pxelinux.cfg/" and place the following line in it (all one line):

"DEFAULT pxelinux.cfg/bzImage initrd=pxelinux.cfg/initrd.img ip=dhcp root=/dev/ram0 init=linuxrc rw".

This line instructs the diskless nodes to use a file named "bzImage" as the kernel and that the initial ramdisk (the file that contains some basic utilities to get the system up and going) will be called "initrd.img". The line will also tell the nodes that their "ip" is obtained via DHCP, that they will be using a file called "linuxrc" as the initial boot script, and that the initial ramdisk will be read/write.

2.4 Step 4: Kernel

Now that the nodes know what their IP is and with what file to bootstrap, they need a kernel to load and a ramdisk (discussed in the next step). The kernel only needs a few drivers compiled in statically (not as modules) so that it can get an ethernet device configured and get an NFS filesystem mounted. If you have never compiled a linux kernel before I recommend reading some of the Linux Kernel documentation first. First, go to the linux kernel source tree "/usr/src/linux-2.4/" (on redhat 9) and type "make menuconfig" to get an interactive menu for configuring the kernel. At this point you basically want to spend some time going through each menu and turning off unnecessary drivers. For example, our machines do not have PCMCIA cards so this driver is disabled. Other drivers like firewire, USB, disk support, etc., will more than likely not be required. The drivers required should be compiled in statically ("*" indicates static while "m" indicates module), not as a module. The two imperative drivers that need to be compiled in statically are “NFS client support” and the driver for the ethernet device being used on the slave for diskless booting. Adding other drivers depends on requirements. Once done getting the custom kernel entirely configured, exit the menu screen by pushing the [esc] key until you are asked if you want to save the configuration file, answering [yes]. At a console, type "make clean && make dep && make -j2 bzImage" and the kernel should begin compiling. If you encounter errors, consult the Linux Kernel documentation. Once the kernel finishes compiling, a file called "bzImage" under "/usr/src/linux-2.4/arch/i386/boot/" should be copied to the "/tftpboot/pxelinux.cfg/" directory. At this point you should have a bootable kernel for the diskless nodes. You may at some point wish to copy the kernel source onto the diskless nodes and do the same process, copying the kernel to /boot along with the System.map, creating the System.map-XXXX symlink (where XXXX is the version, i.e., 2.4.27), and performing the "make modules" and "make modules_install" to put all the modules in place. Doing so will allow you to compile new kernels on a cluster node instead of the head.

2.5 Step 5: Ramdisk

At this point, the diskless nodes are almost ready to boot. A ramdisk image needs to be created in order to provide the diskless nodes with some essential utilities to configure an interface and get NFS going. Most linux systems have a set of ramdisks under /dev/ram0 /dev/ram1 etc. that can be used by the user. Make an ext2 filesystem on ram0 with the following command (redhat 9): "/sbin/mkfs.ext2 /dev/ram0". Make a place to mount the ramdisk under /mnt/ or wherever you prefer with the following command: "mkdir /mnt/ram0". Now try mounting the ramdisk with the following command: "mount -t ext2 /dev/ram0 /mnt/ram0". Typing "df -h" in the console should reveal that that "ram0" has been mounted successfully. Now it's time to start copying over essential utilities and libraries to this ramdisk. The ramdisk is just a miniature filesystem with barebone support. Recall when the "/tftpboot/pxelinux.cfg/default" file was made, we told the diskless nodes to execute a script called "linuxrc", we will begin there. Edit "/mnt/ram0/linuxrc" file and use the following script as a guide for your own custom linuxrc script:

```
---------------------------------------------------------------
#!/bin/bash
# CONFIGURE INTERFACE
/sbin/ifconfig lo 127.0.0.1
/sbin/ifconfig eth0 0.0.0.0

# SET HOSTNAME
/bin/hostname nizzle

# PORTMAP AND RPC.STATD FOR NFS
/sbin/portmap
#/sbin/statd

# GET IP FROM DHCP
/sbin/dhclient -sf /bin/dhcp-adhoc.sh eth0

#
#exec /bin/bash
#

# PIVOT AND BOOT
cd /nfs
/sbin/pivot_root . initrd

# RUN INIT
exec /usr/sbin/chroot . /sbin/init <dev/console >dev/console 2>&1
---------------------------------------------------------------
```

This script is using “bash” (bourne again shell) to configure an interface, set a hostname, start portmap for NFS, and run DHCP to obtain an IP address from the DHCP server. Also commented out is the line "exec /bin/bash" which I recommend uncommenting when you boot the first time. Doing so will allow you to run "ifconfig" from the ramdisk to make sure everything is good up to this point and that networking works. All of the utilities in this script must be on your ramdisk filesystem. Most of these binaries require libraries. One can use the "ldd" command to find out what libraries you need, i.e., "ldd /bin/bash" outputs the following:

```
libtermcap.so.2 => /lib/libtermcap.so.2 (0x40020000)
libdl.so.2 => /lib/libdl.so.2 (0x40024000)
libc.so.6 => /lib/libc.so.6 (0x40027000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

This means that you need these four libraries on your ramdisk under “/lib/” in order for this binary to run. You must "ldd" all your binaries to ensure that you have all the proper library dependencies. I recommend putting the basic utilities (bash, cat, df, hostname, kill, ls, mkdir, mount, ping, ps, ifconfig, chroot, pivot_root, portmap, route, rpc.lockd, statd, dhclient, etc.) on the disk. The default ramdisk size in redhat 9 is 4MB; use it wisely. In the previous "linuxrc" script, there is a line "/sbin/dhclient -sf /bin/dhcp-adhoc.sh eth0" which uses DHCP to configure the interface. I handcrafted a script called dhcp-adhoc.sh to process the DHCP info and apply it to eth0. Below is my custom dhcp-adhoc.sh script:

```
#!/bin/bash
#echo "-- $reason --"
if [ x$reason = xBOUND ]; then
  echo "--- configuring interface [$interface] [$new_ip_address] [$new_host_name] ---";
  /sbin/ifconfig $interface $new_ip_address
  /bin/hostname $new_host_name
  echo "--- mounting nfs (/) ---"
  /bin/mount -t nfs $new_dhcp_server_identifier:/cluster/$new_host_name /nfs
  echo "--- mounting nfs (/bin) ---"
  /bin/mount -t nfs $new_dhcp_server_identifier:/cluster/shared/bin /nfs/bin
  echo "--- mounting nfs (/etc) ---"
  /bin/mount -t nfs $new_dhcp_server_identifier:/cluster/shared/etc /nfs/etc
  echo "--- mounting nfs (/etcmaster) ---"
  /bin/mount -t nfs $new_dhcp_server_identifier:/etc /nfs/etcmaster
  echo "--- mounting nfs (/home) ---"
  /bin/mount -t nfs $new_dhcp_server_identifier:/home /nfs/home
  echo "--- mounting nfs (/lib) ---"
  /bin/mount -t nfs $new_dhcp_server_identifier:/cluster/shared/lib /nfs/lib
  echo "--- mounting nfs (/root) ---"
  /bin/mount -t nfs $new_dhcp_server_identifier:/cluster/shared/root /nfs/root
```

```
  echo "--- mounting nfs (/sbin) ---"
  /bin/mount -t nfs $new_dhcp_server_identifier:/cluster/shared/sbin /nfs/sbin
  echo "--- mounting nfs (/usr) ---"
  /bin/mount -t nfs $new_dhcp_server_identifier:/cluster/shared/usr /nfs/usr
fi
----------------------------------------------------------------
```

This script will obtain the DHCP info, configure the interface, and mount some NFS points for the "pivot_root". I made an "nfs" directory on the root filesystem of the ramdisk called "/nfs/". Under the "/nfs/" directory I stubbed out the basic system directories (bin, sbin, etc, lib, usr ...) and mounted them via "mount" in this shell script.

Referring back to the linuxrc script, I use the "pivot_root" utility to make the "/nfs/" directory the root filesystem. Next, “init” is executed followed by the setting of the root directory to ".". The system should start booting and processing the rc startup scripts. At this point you have a bootable linux system. Everything after this is a matter of tweaking. When testing a ramdisk, go to the “/tftpboot/pxelinux.cfg/” directory and type the following “sync && dd if=/dev/ram0 of=initrd.img”. This will sync the ramdisk to save all the changes made and dump them to the initrd.img file. You can optionally gzip the file if you wish: “gzip initrd.img && mv initrd.img.gz initrd.img”. At a later time you may want to re-edit your ramdisk. This can by accomplished by typing “dd if=initrd.img of=/dev/ram0” and then mounting the disk by typing “mount /dev/ram0 /mnt/ram0”.

2.6 Step 6: Slave Filesystems

This is a good point to start thinking about what you want on the slaves filesystem. Each slave needs a mostly complete linux filesystem to get some basic services up and running so that there is a usable filesystem with which to use. It's up to each individual how they want to accomplish this, but the easiest way to create the slave filesystem is to just copy the root filesystem to an area on the disk dedicated to the cluster, i.e., "/cluster/shared". Next, take all of the root directories and move them to the "/cluster/shared" directory so that they could be used in a shared fashion (i.e., bin, usr, etc, lib, sbin, root) and not "/tmp/" and "/var/" since those are unique to each individual nodes filesystem. Next, make the directories "/cluster/n1/" thru "/cluster/n8/" (or however many nodes being used in the configuration) with the aformentioned set of shared directories just stubbed out: usr, bin, etc, sbin, lib, and root. Looking back on the "dhcp-adhoc.sh" script, the first thing it does is mount the "/cluster/nX" directory where "X" is the number of the node. The next set of actions that take place in the script is the mounting of each of these stubbed out filesystems from the shared directory for the cluster. After all the aforementioned directories are mounted a basic stubbed out filesystem should exist. Having this type of architecture means that anyone can log into any of the nodes, install a package, and the package will be available on each of the nodes as well.

2.7 Step 7: Configuring RC.SYSINIT

Now that the slaves can boot linux via NFS using the custom kernel and ramdisk you created it's time to tweak the filesystem to cater to a diskless environment. One of the first steps to perform is the adjusting of the file "/etc/rc.sysinit", such that any unnecessary commands are not executed during boot. For example, one may not care about USB devices; thus, they should remove the lines performing the USB commands in this script. Additionally, you should make a symlink from “/etc/rc.sysinit” to “/etc/rc.d/rc.sysinit” to avoid the confusion of having two separate versions of these. Anything related to local drive maintenance and so forth should be removed as well. The jist of this process is to examine all of the [Fail] messages on bootup and find out why a [Fail] message appears, and then resolve the problem by adjusting or removing the commands causing the failure. Most [Fail] messages occur because this script is attempting to either use a utility that requires a module that is not present in the kernel or because the script is trying to do something to a local disk drive that is not appropriate. Expect to boot the node a few times in order to get this script configured well. The configuration process depends on the system, hardware, and environment being used. There is no real turn-key solution for this process yet. Once this arduous process has been completed, one should be able to boot the nodes cleanly and get a linux login screen. See the appendix for an example.

2.8 Step 8: Post-Install Notes

While the ramdisk is booting, you may encounter nfs/lockd messages being spammed to the console because portmap and/or rpc.statd is not running. I have found that you need to download the source RPM from the redhat 9 source and use that "statd" binary not the "rpc.statd" binary that comes with the system. The same goes for the rpc.statd binary on each of the nodes filesystems, use the "statd" binary, not the "rpc.statd" binary that comes with the system. You could make a symlink from "statd" to "rpc.statd" if you like. Also check your "/etc/hosts.allow" and try adding the following line to fix things: "portmap : 192.168.1.0/255.255.255.0 : allow".

It is important that each of the nodes have their own "/var/" and "/tmp/" directories since they do their own accounting in them (assuming you want write access on the filesystems).

If you would like to share password files between the master and the slaves, I made a "/etcmaster" and mounted that in the ramdisk phase and pointed the slaved "/etc/passwd", "/etc/shadow", and "/etc/group" to the respective locations on "/etcmaster/". This will allow the administrator to make a user on the master node, which automatically gets created on the slaves as well.

You may also wish to stub out an "/etc/mtab" to have a fake entry for your NFS mount so that when the "df" command is used it reports a live filesystem (although fake). This will allow one to get an idea of how much space is available via the NFS. An example would be 192.168.1.254:/ / nfs rw,bg,soft,intr,addr=192.168.1.254 0 0.

Having a simple way to login to each of the nodes can be important. I chose to setup "rsh" on each one of those machines so that anybody but root could log into a node by simply typing "rsh n1" or the node of choice from the master. This was accomplished by enabling rsh under the /etc/xinetd.d/rsh file (assuming you installed rsh) and enabling it by changing "disable = yes" to "disable = no". Do not forget to update the "/etc/hosts.allow" to permit anyone on the subnet to rsh into the box: "in.rshd : 192.168.1."

Having a method of invoking a command on all the machines is important. Not many people want to log into each node to start up a process. The following is a shell script* called "crun" that automates this process:

```
#!/bin/sh
# $Id: crun,v 1.2 2004/08/04 13:08:32 erikg Exp $
# Erik Greenwald <erikg@arl.army.mil> 3-6255 blah blah
# get this from dhcp? or a heartbeat monitor?
hosts="n1 n2 n3 n4 n5 n6 n7 n8"

# default parms
verbose="no"
user=`whoami`
background="no"

while [ `echo $1 | cut -b 1` = '-' ]
do
        parm=`echo $1 | cut -b 2`
        shift
        case "$parm" in
                v)
                        verbose=yes
                        ;;
                l)
                        user=$1
                        preflag="-l $1"
                        shift
                        ;;
                b)
                        background=yes
                        preflag="$preflag -n"
                        postflag='>/dev/null 2>/dev/null &'
                        ;;
                h)
                        echo "Usage:"
                        echo "          crun [-v] [-b] [-h] -[l user] command"
                        echo
```

*Developed by Erik Greenwald, U.S. Army Research Laboratory.

```
                echo " -v           verbose mode"
                echo " -b           background mode"
                echo " -h           show this help"
                echo " -l user       rsh as user"
                                        echo
                                        exit
                                        ;;
                        *)
                                        echo "Usage:"
                                        echo "          crum [-v] [-b] [-h] [-l user] command"
                                        echo
                                        echo "try -h for help"
                                        echo
                                        exit -1
                                        ;;
        esac
done

for host in $hosts
do
        if [ $verbose = "yes" ]
        then
                echo "rsh $preflag $host \"$* $postflag\"" 1>&2
        fi
        rsh $preflag $host "$* $postflag"
done

exit
```

Invoking the script like this, "crun w", will show you who is logged in on all the boxes, while "crun -b my.daemon" will start and background a daemon on all the machines.

I wrote a script that uses the “crun” script that is useful for rebooting all the nodes called "creboot".

```
#!/bin/bash
echo "Rebooting all nodes, please wait... "
crun -b reboot -f
echo "Complete."
```

Because the nodes are not fully equipped to shutdown properly, I force them to reboot without shutting down entirely. This is due to the way in which redhat 9 has the “rc” scripts configured. It is up to the administrator and his needs to configure the nodes to shutdown and reboot without forcing a fast reboot.

It is useful to have a “/etc/hosts” file on the master and on the shared “/etc” of the nodes that contains the host and IP address of each of the machines in the cluster. Then, facilities like “rsh” can be used with just the hostname instead of the IP address, i.e., “rsh n1” instead of “rsh 192.168.1.1”.

The following is the modified version of “/etc/rc.sysinit”.

FILESYSTEM TOUR:

The following is series of “ls” outputs to get a better idea of where all these aforementioned directories are and what's inside of them:

```
[user@master:/]$ ls
bin  cluster  dev  home      initrd  lost+found  mnt  proc  sbin     tmp  var
boot  d       etc  infiniband  lib    misc       opt  root  tftpboot  usr

[user@master:/cluster]$ ls
n1  n2  n3  n4  n5  n6  n7  n8  shared

[user@master:/cluster/shared]$ ls
bin  etc  lib  root  sbin  usr

[user@master:/tftpboot]$ ls
pxelinux.0  pxelinux.cfg

[user@machine:/tftpboot/pxelinux.cfg]$ ls
bzImage  default  initrd.img

[user@machine:/cluster/n1]$ ls
bin  boot  dev  etc  etcmaster  home  initrd  lib  node  proc  root  sbin  tmp  usr  var
```

2.9 Conclusion

While trying to keep this report short and complete it is difficult to cover every aspect in the greatest of detail. I recommend using a search engine to seek out answers to issues you may run into due to differing hardware platforms, linux versions, and requirements in general. The previous work section contains some useful links that cover other various details and configuration ideas not mentioned here. Allow for at least a full day to configure the cluster as this is not a turn-key solution. The appendix shows the final working version of our modified rc.sysinit script.

Appendix. /etc/rc.d/rc.sysinit

```
---------------------------------------------------------------
#!/bin/bash
#
#/etc/rc.d/rc.sysinit - run once at boot time
#
# Taken in part from Miquel van Smoorenburg's bcheckrc.
#
# Rerun ourselves through initlog
if [ -z "$IN_INITLOG" -a -x /sbin/initlog ]; then
   exec /sbin/initlog $INITLOG_ARGS -r /etc/rc.d/rc.sysinit
fi

# If we're using devfs, start devfsd now - we need the old device names
[ -e /dev/.devfsd -a -x /sbin/devfsd ] && /sbin/devfsd /dev

HOSTNAME=`/bin/hostname`
if [ -f /etc/sysconfig/network ]; then
   . /etc/sysconfig/network
else
   NETWORKING=no
fi
if [ -z "$HOSTNAME" -o "$HOSTNAME" = "(none)" ]; then
   HOSTNAME=localhost
fi

. /etc/init.d/functions

# Start the graphical boot, if necessary
if [ "$BOOTUP" = "graphical" ]; then
  if [ -x /usr/bin/rhgb ]; then
     /usr/bin/rhgb
  else
     export BOOTUP=color
  fi
fi

last=0
for i in `LC_ALL=C grep '^[0-9]*.*respawn:/sbin/mingetty' /etc/inittab | sed 's/^.* tty\([0-9][0-
9]*\).*/\1/g'`; do
      > /dev/tty$i
      last=$i
done
```

```
if [ $last -gt 0 ]; then
    > /dev/tty$((last+1))
    > /dev/tty$((last+2))
fi

if [ "`/sbin/consoletype`" = "vt" -a -x /sbin/setsysfont ]; then

  echo -n "Setting default font ($SYSFONT): "
  /sbin/setsysfont
  if [ $? -eq 0 ]; then
    success
  else
    failure
  fi
  echo ; echo
fi

# Print a text banner.
echo -en $"\t\tWelcome to "
if LC_ALL=C grep -q "Red Hat" /etc/redhat-release ; then
 [ "$BOOTUP" = "color" ] && echo -en "\\033[0;31m"
 echo -en "Red Hat"
 [ "$BOOTUP" = "color" ] && echo -en "\\033[0;39m"
 PRODUCT=`sed "s/Red Hat \(.*\) release.*/\1/" /etc/redhat-release`
 echo " $PRODUCT"
else
 PRODUCT=`sed "s/ release.*//g" /etc/redhat-release`
 echo "$PRODUCT"
fi
if [ "$PROMPT" != "no" ]; then
 echo -en $"\t\tPress 'I' to enter interactive startup."
 echo
 sleep 1
fi

# Fix console loglevel
/bin/dmesg -n $LOGLEVEL

# Mount /proc (done here so volume labels can work with fsck)
action $"Mounting proc filesystem: " mount -n -t proc /proc /proc

# Configure kernel parameters
action $"Configuring kernel parameters: " sysctl -e -p /etc/sysctl.conf
```

```
# Set the system clock.
ARC=0
SRM=0
UTC=0

if [ -f /etc/sysconfig/clock ]; then
  . /etc/sysconfig/clock

  # convert old style clock config to new values
  if [ "${CLOCKMODE}" = "GMT" ]; then
    UTC=true
  elif [ "${CLOCKMODE}" = "ARC" ]; then
    ARC=true
  fi
fi

CLOCKDEF=""
CLOCKFLAGS="$CLOCKFLAGS --hctosys"

case "$UTC" in
  yes|true)
   CLOCKFLAGS="$CLOCKFLAGS --utc";
   CLOCKDEF="$CLOCKDEF (utc)";
  ;;
  no|false)
   CLOCKFLAGS="$CLOCKFLAGS --localtime";
   CLOCKDEF="$CLOCKDEF (localtime)";
  ;;
esac

case "$ARC" in
    yes|true)
                CLOCKFLAGS="$CLOCKFLAGS --arc";
                CLOCKDEF="$CLOCKDEF (arc)";
    ;;
esac
case "$SRM" in
    yes|true)
                CLOCKFLAGS="$CLOCKFLAGS --srm";
                CLOCKDEF="$CLOCKDEF (srm)";
    ;;
esac

/sbin/hwclock $CLOCKFLAGS

action $"Setting clock $CLOCKDEF: `date`" date
```

```
if [ "`/sbin/consoletype`" = "vt" -a -x /bin/loadkeys ]; then
 KEYTABLE=
 KEYMAP=
 if [ -f /etc/sysconfig/console/default.kmap ]; then
  KEYMAP=/etc/sysconfig/console/default.kmap
 else
  if [ -f /etc/sysconfig/keyboard ]; then
    . /etc/sysconfig/keyboard
  fi
  if [ -n "$KEYTABLE" -a -d "/lib/kbd/keymaps" ]; then
     KEYMAP=$KEYTABLE
  fi
 fi
 if [ -n "$KEYMAP" ]; then
  # Since this takes in/output from stdin/out, we can't use initlog
  if [ -n "$KEYTABLE" ]; then
    echo -n $"Loading default keymap ($KEYTABLE): "
  else
    echo -n $"Loading default keymap: "
  fi
  loadkeys $KEYMAP < /dev/tty0 > /dev/tty0 2>/dev/null && \
     success $"Loading default keymap" || failure $"Loading default keymap"
  echo
 fi
fi

# Set the hostname.
action $"Setting hostname ${HOSTNAME}: " hostname ${HOSTNAME}

if [ -f /fastboot ] || LC_ALL=C grep -iq "fastboot" /proc/cmdline 2>/dev/null ; then
                fastboot=yes
fi

# LVM initialization
if [ -f /etc/lvmtab -a ! -e /proc/lvm ] ; then
       modprobe lvm-mod >/dev/null 2>&1
fi
if [ -e /proc/lvm -a -x /sbin/vgchange -a -f /etc/lvmtab ]; then
       action $"Setting up Logical Volume Management:" /sbin/vgscan && /sbin/vgchange -a y
fi

# Clear mtab
>/etc/mtab

# Remove stale backups
rm -f /etc/mtab~ /etc/mtab~~
```

```
# The root filesystem is now read-write, so we can now log
# via syslog() directly..
if [ -n "$IN_INITLOG" ]; then
   IN_INITLOG=
fi

if ! LC_ALL=C grep -iq nomodules /proc/cmdline 2>/dev/null && [ -f /proc/ksyms ]; then
   USEMODULES=y
fi

#XXX
# Our modutils don't support it anymore, so we might as well remove
# the preferred link.
rm -f /lib/modules/preferred /lib/modules/default
if [ -x /sbin/depmod -a -n "$USEMODULES" ]; then
   # If they aren't using a recent sane kernel, make a link for them
   if [ ! -n "`uname -r | LC_ALL=C grep -- "-"`" ]; then
      ktag="`cat /proc/version`"
      mtag=`LC_ALL=C grep -l "$ktag" /lib/modules/*/.rhkmvtag 2> /dev/null`
      if [ -n "$mtag" ]; then
         mver=`echo $mtag | sed -e 's,/lib/modules/,,' -e 's,/.rhkmvtag,,' -e 's,[       ].*$,,'`
      fi
      if [ -n "$mver" ]; then
        ln -sf /lib/modules/$mver /lib/modules/default
      fi
   fi
   if [ -L /lib/modules/default ]; then
                INITLOG_ARGS= action $"Finding module dependencies: " depmod -A default
   else
                INITLOG_ARGS= action $"Finding module dependencies: " depmod -A
   fi
fi

if [ -f /proc/sys/kernel/modprobe ]; then
  if [ -n "$USEMODULES" ]; then
      sysctl -w kernel.modprobe="/sbin/modprobe" >/dev/null 2>&1
      sysctl -w kernel.hotplug="/sbin/hotplug" >/dev/null 2>&1
  else
      # We used to set this to NULL, but that causes 'failed to exec' messages"
      sysctl -w kernel.modprobe="/bin/true" >/dev/null 2>&1
      sysctl -w kernel.hotplug="/bin/true" >/dev/null 2>&1
  fi
fi
```

```
# Load modules (for backward compatibility with VARs)
if [ -f /etc/rc.modules ]; then
        /etc/rc.modules
fi

if [ -x /sbin/devlabel ]; then
        /sbin/devlabel restart
fi

# Configure machine if necessary.
if [ -f /.unconfigured ]; then
   if [ "$BOOTUP" = "graphical" ]; then
        chvt 1
   fi

   if [ -x /usr/bin/passwd ]; then
      /usr/bin/passwd root
   fi
   if [ -x /usr/sbin/netconfig ]; then
        /usr/sbin/netconfig
   fi
   if [ -x /usr/sbin/timeconfig ]; then
        /usr/sbin/timeconfig
   fi
   if [ -x /usr/sbin/kbdconfig ]; then
        /usr/sbin/kbdconfig
   fi
   if [ -x /usr/sbin/authconfig ]; then
        /usr/sbin/authconfig --nostart
   fi
   if [ -x /usr/sbin/ntsysv ]; then
        /usr/sbin/ntsysv --level 35
   fi

   # Reread in network configuration data.
   if [ -f /etc/sysconfig/network ]; then
        . /etc/sysconfig/network

        # Reset the hostname.
        action $"Resetting hostname ${HOSTNAME}: " hostname ${HOSTNAME}
   fi

   rm -f /.unconfigured
fi
```

```
# Clean out /.
rm -f /fastboot /fsckoptions /forcefsck /.autofsck /halt /poweroff

# Do we need (w|u)tmpx files? We don't set them up, but the sysadmin might...
_NEED_XFILES=
[ -f /var/run/utmpx -o -f /var/log/wtmpx ] && _NEED_XFILES=1

# Clean up /var.  I'd use find, but /usr may not be mounted.
for afile in /var/lock/* /var/run/* ; do
                if [ -d "$afile" ]; then
                  case "`basename $afile`" in
                                news|mon)   ;;
                                sudo)
                                                        rm -f $afile/*/* ;;
                                *)                      rm -f $afile/* ;;
                  esac
                else
                  rm -f $afile
                fi
done
rm -f /var/lib/rpm/__db*

# Reset pam_console permissions
[ -x /sbin/pam_console_apply ] && /sbin/pam_console_apply -r

{
# Clean up utmp/wtmp
>/var/run/utmp
touch /var/log/wtmp
chgrp utmp /var/run/utmp /var/log/wtmp
chmod 0664 /var/run/utmp /var/log/wtmp
if [ -n "$_NEED_XFILES" ]; then
  >/var/run/utmpx
  touch /var/log/wtmpx
  chgrp utmp /var/run/utmpx /var/log/wtmpx
  chmod 0664 /var/run/utmpx /var/log/wtmpx
fi

# Delete X locks
rm -f /tmp/.X*-lock

# Delete VNC & X locks
rm -rf /tmp/.X*-unix

# Delete ICE locks
rm -rf /tmp/.ICE-unix
```

```
# Delete Postgres sockets
rm -f /tmp/.s.PGSQL.*

# Initialize the serial ports.
if [ -f /etc/rc.serial ]; then
                . /etc/rc.serial
fi

# Boot time profiles. Yes, this should be somewhere else.
if LC_ALL=C grep -q "netprofile=" /proc/cmdline ; then
   cmdline=`cat /proc/cmdline`
   for arg in $cmdline ; do
     if [ "${arg##netprofile=}" != "${arg}" ]; then
       [ -x /usr/sbin/redhat-config-network-cmd ] &&
                /usr/sbin/redhat-config-network-cmd --profile ${arg##netprofile=}
     fi
   done
fi

# Generate a header that defines the boot kernel.
#/sbin/mkkerneldoth

# Adjust symlinks as necessary in /boot to keep system services from
# spewing messages about mismatched System maps and so on.
if [ -L /boot/System.map -a -r /boot/System.map-`uname -r` -a \
                ! /boot/System.map -ef /boot/System.map-`uname -r` ]; then
                ln -s -f System.map-`uname -r` /boot/System.map
fi
if [ ! -e /boot/System.map -a -r /boot/System.map-`uname -r` ]; then
                ln -s -f System.map-`uname -r` /boot/System.map
fi

# The special Red Hat kernel library symlink must point to the right library
# We need to deal with cases where there is no library, and we need to
# deal with any version numbers that show up.
shopt -s nullglob
for library in /lib/kernel/$(uname -r)/libredhat-kernel.so* ; do
                ln -f $library /lib/
                ldconfig -n /lib/
done
shopt -u nullglob

# Now that we have all of our basic modules loaded and the kernel going,
# let's dump the syslog ring somewhere so we can find it later
dmesg -s 131072 > /var/log/dmesg
(/bin/date;
```

```
 /bin/uname -a;
 /bin/cat /proc/cpuinfo;
 [ -r /proc/modules ] && /bin/cat /proc/modules;
 [ -r /proc/ksyms ] && /bin/cat /proc/ksyms) >/var/log/ksyms.0
sleep 1
kill -TERM `/sbin/pidof getkey` >/dev/null 2>&1
} &
if [ "$PROMPT" != "no" ]; then
   /sbin/getkey i && touch /var/run/confirm
fi
wait
```

NO. OF
COPIES ORGANIZATION

1 (PDF ONLY) DEFENSE TECHNICAL INFORMATION CTR
DTIC OCA
8725 JOHN J KINGMAN RD
STE 0944
FORT BELVOIR VA 22060-6218

1 US ARMY RSRCH DEV & ENGRG CMD
SYSTEMS OF SYSTEMS INTEGRATION
AMSRD SS T
6000 6TH ST STE 100
FORT BELVOIR VA 22060-5608

1 INST FOR ADVNCD TCHNLGY
THE UNIV OF TEXAS
AT AUSTIN
3925 W BRAKER LN STE 400
AUSTIN TX 78759-5316

1 DIRECTOR
US ARMY RESEARCH LAB
IMNE ALC IMS
2800 POWDER MILL RD
ADELPHI MD 20783-1197

3 DIRECTOR
US ARMY RESEARCH LAB
AMSRD ARL CI OK TL
2800 POWDER MILL RD
ADELPHI MD 20783-1197

3 DIRECTOR
US ARMY RESEARCH LAB
AMSRD ARL CS IS T
2800 POWDER MILL RD
ADELPHI MD 20783-1197

ABERDEEN PROVING GROUND

1 DIR USARL
AMSRD ARL CI OK TP (BLDG 4600)

NO. OF
COPIES ORGANIZATION

1 DIRECTOR FORCE DEV
DAPR FDZ
RM 3A522
460 ARMY PENTAGON
WASHINGTON DC 20310-0460

1 US ARMY TRADOC ANL CTR
ATRC W
A KEINTZ
WSMR NM 88002-5502

1 USARL
AMSRD ARL SL EA
R FLORES
WSMR NM 88002-5513

1 USARL
AMSRD ARL SL EI
J NOWAK
FORT MONMOUTH NJ 07703-5601

ABERDEEN PROVING GROUND

1 US ARMY DEV TEST COM
CSTE DTC TT T
APG MD 21005-5055

1 US ARMY EVALUATION CTR
CSTE AEC SVE
R BOWEN
4120 SUSQUEHANNA AVE
APG MD 21005-3013

1 US ARMY EVALUATION CTR
CSTE AEC SVE S
R POLIMADEI
4120 SUSQUEHANNA AVE
APG MD 21005-3013

1 US ARMY EVALUATION CTR
CSTE AEC SV L
R LAUGHMAN
4120 SUSQUEHANNA AVE
APG MD 21005-3013

8 DIR USARL
AMSRD ARL SL
J BEILFUSS
AMSRD ARL SL B
M PERRY

NO. OF
COPIES ORGANIZATION

AMSRD ARL SL BB
D FARENWALD
C HUNT
AMSRD ARL SL BE
L ROACH
J SHUMACHER (3 CPS)

INTENTIONALLY LEFT BLANK.